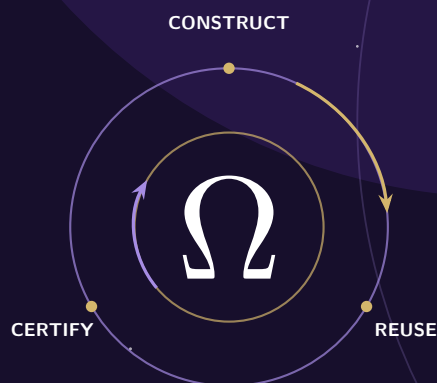


# AGI ALPHA FLYWHEEL

From Completed Work  
to Compounding Capability

Transformational acceleration study / Edition 02



**MAKE THE NEXT CYCLE CHEAPER.**

SUCCESSOR  $\Omega$  HARNESS  
**ACTIVATE. EVOLVE. SCALE.**

Powered by TRANSFORMATIONAL INTELLIGENCE  $\infty$   
and the applicable \$AGIALPHA utility.

Vincent Boucher

Montréal, Québec, Canada / 24 October 2026

Finite source execution. Original research edition and negative findings preserved.

## Abstract

Can accepted work change the means of producing the next result, with a benefit after construction, validation and upkeep? We test this question through the released AGI ALPHA credential-free workbench. A finite controller compares exact methods, constructs reusable dependency-component bases, checks fresh results independently and retains or rejects candidate means. Six matched evaluation windows preserve the requirement mix; a conventional adaptive cache and preauthored expert receive the same information, methods and budgets. All four arms qualify 864/864 registered tasks. The flywheel and conventional adaptive cache tie at 2,887,585 counted units; the authored expert uses 2,117,413, or 26.7% less. Replacing the fixed full-mask instrument reduces counted work by 96.9%, but isolated basis retention saves only 0.55% in the registered removal test. All eight streams pass the amortized productivity-growth criterion and fail the stricter acceleration criterion. Capabilities saturate after initial acquisition; self-accelerating capability construction is not demonstrated. The contribution is executable attribution and negative evidence in a finite synthetic mission family. Declared computational counts exclude unmeasured authoring labor. Frozen protocols, raw receipts, failures and a runnable browser companion accompany the paper.

### A claim the next run can disprove

Does a retained executable transformation improve qualified output per total computational work on fresh, matched requirements, after its own construction and validation are charged?

**Continuity.** Research edition01 and its conventional-cache tie, expert lead, rejected descendant, annotation failure and reuse/rebuild crossover remain unchanged. This separately frozen extension preserves that source and its authoritative PDF [1].

## The result in one page

### Useful transformation. Modest inheritance benefit. No sustained acceleration.

The finite loop runs and retains executable means. The large saving against the fixed instrument comes mainly from conventional method replacement. The isolated retention benefit is small; the strongest authored workflow remains cheaper. The new evidence does not establish a self-accelerating construction process.

| Workflow                    | Qualified | Counted work | Work / qualified |
|-----------------------------|-----------|--------------|------------------|
| Flywheel                    | 864/864   | 2,887,585    | 3,342.11         |
| Conventional cache          | 864/864   | 2,887,585    | 3,342.11         |
| Authored expert             | 864/864   | 2,117,413    | 2,450.71         |
| Frozen full-mask instrument | 864/864   | 93,849,181   | 108,621.74       |

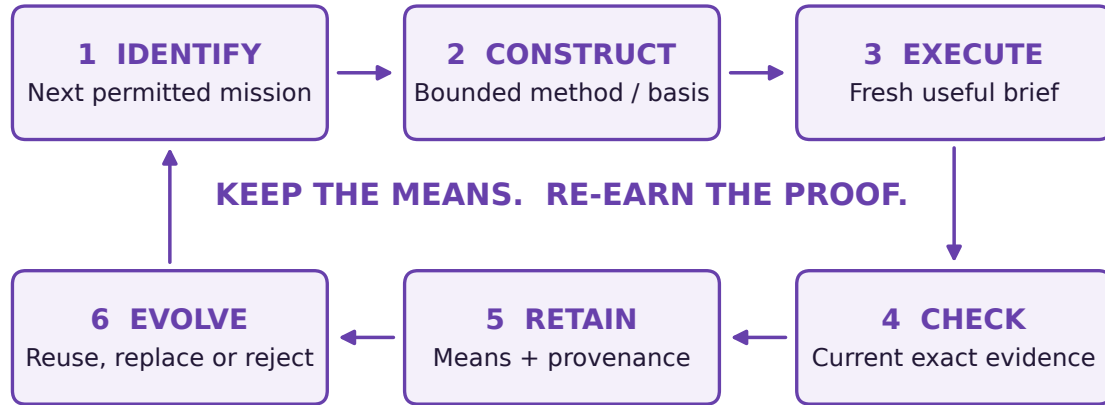
**What changed.** Bounded exact selection and current checkable component frontiers replace repeated full-mask traversal. Four exact component bases are constructed, retained and executed on later tasks with fresh weights. Every fresh answer is revalidated; production authority stays empty.

**What did not follow.** The conventional cache ties exactly. The expert spends 26.7% fewer counted units. The registered archive-removal test attributes only 1,995 units, or 0.55%, to persistence of component bases. All four bases arrive in the first window; later windows plateau. None of eight streams passes the strict acceleration criterion, and no multi-asset reinvestment chain is executed.

**Use it now.** Open the self-contained browser companion and choose Try it now. It executes a separate six-mission teaching stream, shows useful reuse and a refused incompatible basis, and lets the reader export a brief, the actual reusable program and a recoverable checkpoint. Recorded confirmation and fresh in-tab computation are labeled separately.

**Read the evidence.** The complete source package includes 3,456 primary observations, 192 workbench packs, the frozen protocol, failures, independent checking, actual load and recovery records, editable vector figures and the unchanged original edition.

## A finite recurring loop through the released workbench



Fixed objective • Fixed evaluation • Finite budget • Empty production authority

**WAIT, retain incumbent and STOP are valid outcomes.**

**Figure 1:** Accepted work can leave useful means. Every next result still needs current proof. The outside boundary stays fixed.

## 1 The flywheel has to change what the next cycle can do

An accepted result has two possible lives. It can answer the present question, and it can leave a means of answering a later question. The second life is the subject of this paper. A useful archive contains more than evidence that work once succeeded: it contains a program, instrument or representation that can be executed under a fresh requirement, with a new check of its applicability. Retaining a name, a favorable score or an authority token does not establish that property.

We study a finite family of evidence-preserving briefs. Given supplied source clauses, a nonnegative weight for each clause and explicit prerequisites, produce a brief of at most  $k$  clauses that maximizes supported weight without dropping a prerequisite. A useful answer to one instance is just a selected set. A useful means of production is a reusable representation of the feasible selections for a recurring dependency structure. The latter can serve a different document, different weights and a different permitted length. Its local-position schema also supports identifier rebinding, but the registered primary keeps the same identifier vocabulary; it does not test that additional generalization.

The aim is deliberately concrete: make the next accepted unit of work cheaper, or expand the work that qualifies under a fixed budget. The study does not measure whether a reader understands a document, whether an annotation is factually correct, or whether customers will pay for the answer. Those are separate questions requiring separate evidence.

### 1.1 Growth, productivity and acceleration are different endpoints

Let  $\mathcal{T} = \{t_1, \dots, t_N\}$  be a stream fixed before evaluation. Let  $q_i \in \{0, 1\}$  denote qualification under the frozen task criterion, and let  $c_i$  include the work attributable to obtaining and checking that result. Acquisition and upkeep are charged even when they fail to improve an answer. For a total resource cap  $B$ , define

$$Q(B) = \sum_{i=1}^N q_i \mathbf{1} \left\{ \sum_{j \leq i} c_j \leq B \right\}. \quad (1)$$

Tasks stopped before execution remain in the denominator with zero qualification. A population-level claim would require an appropriate task distribution; here  $\mathcal{T}$  is an explicitly constructed finite distribution.

For a matched task window  $g$ , define computational productivity  $p_g = Q_g/C_g$ , where  $C_g$  includes acquisition, selection, failed attempts, current checking, retention and the online independent Python audit. Comparison with a fixed-instrument control is normalized by the same window’s difficulty. A decreasing cost per qualified task is a speedup. An increasing speedup is a further claim. An increasing rate of improvement is a further claim again. A large early saving followed by a plateau does not establish a rising improvement rate.

| Claim                            | Evidence required in this study                                                                                                                |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Recurring autonomous execution   | One finite launch selects later permitted missions, executes them and stops without a human triggering each phase.                             |
| Useful capability inheritance    | A later result reads and executes a prior admitted asset; removal with reconstruction changes charged work or qualification.                   |
| Amortized computational benefit  | The saving survives acquisition, fresh checks, failures, upkeep and audit over the registered horizon.                                         |
| Self-accelerating construction   | Earlier means causally lower the full cost of constructing a genuinely different later means; the saved resource supports further useful work. |
| Rising improvement rate          | The separately frozen acceleration statistic passes, rather than merely a first-window speedup.                                                |
| Operational or commercial growth | Sustained service or actual demand and economic outcomes; neither follows from this finite synthetic experiment.                               |

**Table 1:** A claim ladder. The implemented loop and the measured growth verdict must be reported separately.

## 1.2 What TRANSFORMATIONAL INTELLIGENCE changes here

The foundational manuscript treats the representation, available operators, verification procedure and interface as objects that may be redesigned while preserving a purpose [2]. We instantiate that idea at a specific obstruction. In the preserved first flywheel study, terminal validation consumes 1,053,126 of 2,017,944 declared work units, or 52.2%. Across acquisition, qualification and validation, 480 all-method comparison calls account for 86.96% of the declared total. These are the original study’s accounting categories, not a new CPU profile. The diagnostic is reproduced in `foundations/v1-bottleneck-audit.json`.

The transformation preserves the task’s objective and admissible outputs. It changes how the system represents the feasible set and how it supplies current evidence. The original workbench need not compare every available method to validate every result. A bounded enumerator can exploit the clause cap. A complete local basis can factor a separable dependency graph. A fresh certificate can expose the exact intermediate quantities that justify an answer. None of these changes relaxes the exact structural endpoint.

This is a transformation within an author-supplied grammar. The code does not invent the concept of dynamic programming, discover arbitrary algorithms or rewrite its evaluator. The bounded-cardinality idea already appears in the project’s native source; the present experiment transfers and tests it through the credential-free browser/workbench seam. The empirical question is whether this supplied transformation becomes useful retained means under full accounting.

## 1.3 A means of production, rather than a remembered answer

Suppose a later source has the same small prerequisite motif as an earlier source but changes which claims are valuable. Reusing the earlier winning selection can be wrong. A complete feasible basis instead says which local selections are allowed. The new mission recomputes values on that basis using the new weights, then combines local frontiers under its new cap. Thus the reusable object represents a family of permissible future actions. It contains no current winner, proof grant or production authority.

This distinction connects to program-library construction and executable skill reuse [9, 10]. The present study’s narrower contribution is a complete cost and authority audit of a finite recurring loop, including a strong conventional cache with the same means. Equal cache performance is a legitimate result. It prevents a conventional reuse mechanism from being relabeled as unique intelligence. Cost-aware agent evaluation and negative transfer motivate reporting failed acquisition and the cases in which a simpler route wins [11, 12].

| Inspectable operation | Concrete contribution                                                                                                                 |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Identify              | Select the next due request from the finite manifest; expose its objective, scope and remaining budget.                               |
| Out-Learn             | Measure the declared work of candidate production/checking methods on separate development inputs.                                    |
| Out-Think             | Recognize that a cap and weak dependency components change the feasible search space.                                                 |
| Out-Design            | Construct a typed complete local basis and an explicit mapping to current source identifiers.                                         |
| Out-Strategise        | Select an available method using the fixed policy; charge acquisition and retain an incumbent when a descendant is unhelpful.         |
| Out-Execute           | Use the released workbench, emit the current result and certificate, and validate them before accepting a research result.            |
| Out-Evolve            | Retain admitted executable data with provenance, test its contribution on later work, and stop or retire it under the declared rules. |

**Table 2:** The six Out-stage foundations become auditable operations, not seven claims of independently learned intelligence [3–8].

## 2 A checkable transformation of production

### 2.1 The fixed structural objective

Let  $V$  be the source clauses,  $E$  the directed prerequisite edges and  $w_i \geq 0$  an integer weight. The input is an annotated acyclic graph. A set  $S \subseteq V$  is feasible when  $|S| \leq k$  and every prerequisite of every selected node is also in  $S$ . Checking the direct prerequisite relation for every selected node implies transitive closure. The objective is

$$\max_{S \subseteq V} \sum_{i \in S} w_i \quad \text{subject to} \quad |S| \leq k, \quad i \in S \implies \text{parents}(i) \subseteq S. \quad (2)$$

The implementation also enforces source membership, unique selected identifiers and exact integer arithmetic over the registered range. Equal optima may have different witnesses; qualification concerns feasibility and optimum value, with the producer’s tie rule separately documented.

### 2.2 U1: use the cap before generating candidates

The fixed instrument traverses all  $2^n$  masks. The bounded instrument traverses only subsets with cardinality at most  $k$ . For the registered  $n = 12$  and  $k \in \{1, 2, 3\}$ , that means 13, 79 or 299 subsets before dependency checks, instead of 4,096 masks. These combinatorial counts explain an opportunity; they do not replace measured full-run cost. Both the producer and its current verifier still perform actual checks, and the independent Python audit is charged. The conventional and expert workflows receive this method from launch.

### 2.3 U2: retain a complete local production basis

Decompose the undirected version of the prerequisite graph into weak components  $C_1, \dots, C_m$ . For each component of size at most four, construct

$$\mathcal{B}_j(a) = \{S \subseteq C_j : |S| = a, S \text{ is dependency-closed}\}, \quad 0 \leq a \leq 3. \quad (3)$$

The asset serializes all such masks, ordered local dependencies, cap support, the source identity and a digest. Local positions are defined by sorted current identifiers; the source binding and dependency representation must match exactly. No graph-isomorphism oracle or free semantic matching is assumed. Components larger than four fall back to bounded exact selection. That fallback is a required failure boundary, not a discarded inconvenient task.

For each current task, the producer evaluates

$$f_j(a) = \max_{S \in \mathcal{B}_j(a)} \sum_{i \in S} w_i, \quad (4)$$

$$F_0(0) = 0, \quad F_0(a > 0) = -\infty, \quad (5)$$

$$F_j(a) = \max_{0 \leq b \leq a} \{F_{j-1}(a-b) + f_j(b)\}. \quad (6)$$

The best objective is  $\max_{a \leq k} F_m(a)$ . Infeasible frontier cells are serialized as JSON `null`, not as an unportable infinity. Attaining local masks and a reconstructed global selection accompany the values. A current certificate includes the graph, weight and source bindings, cap, selected identifiers, local frontiers and every global recurrence row.

**Proposition 1** (Preservation of the structural objective). *If the weak components partition the current graph, every  $\mathcal{B}_j(a)$  is complete for its admitted cap, and the current frontiers and recurrence are correct, the reconstructed feasible selection achieves the same optimum as global subset enumeration.*

*Proof.* Every feasible global selection intersects each weak component in a feasible local selection. Conversely, a union of feasible local selections is dependency-closed because no edge crosses weak components; it is globally feasible exactly when the cardinalities sum to at most  $k$ . Additive weights make the objective the sum of local objectives. Induction over  $j$  establishes that  $F_j(a)$  is the best value for exactly  $a$  selected nodes from the first  $j$  components. Maximizing over  $a \leq k$  gives the global optimum. Attaining witnesses supply a feasible set with that value.  $\square$

The proof is conditional on supplied annotations and complete bases. It is not a proof that the source is true. The verifier therefore reconstructs components from the current source, checks basis completeness, recalculates current local optima and checks the recurrence. A separate Python implementation enumerates combinations without importing the candidate’s JavaScript selector. Candidate output is data; it cannot replace that checker through the supported interface. Same-workspace file hashes detect changed bytes, but do not constitute hostile operating-system isolation.

## 2.4 Retention needs a balance sheet

Let  $A$  be an acquired representation,  $C_{\text{acquire}}(A)$  its construction and admission cost, and  $u_i(A)$  the later reduction in work relative to an allowed reconstruction or replacement. Retention has an observed finite computational benefit only if

$$\sum_i u_i(A) > C_{\text{acquire}}(A) + C_{\text{retain}}(A) + C_{\text{repair}}(A) + C_{\text{audit}}(A), \quad (7)$$

at equal or better qualification and under the same information. These are computational units defined by the protocol. Unknown authoring labor cannot be silently substituted with zero. The comparison is path-dependent: a more general old instrument may cost more than a newly specialized one, and fresh checking can dominate a nominally cheap lookup.

There is no perpetual-growth theorem here. For a fixed finite benchmark with bounded productivity, improvements can saturate. The foundational TI manuscript itself does not permit infinitely many fixed positive gains under a bounded objective. A useful flywheel must be able to retain the incumbent, wait, reject a descendant, rebuild a tool and stop. Those options are part of successful bounded operation.

## 3 A prospective experiment with a strong conventional control

### 3.1 Fixed tasks, methods, information and stopping rules

The extension uses eight complete streams, six windows per stream and 18 terminal tasks per window. Every window contains one task for each of six topology regimes crossed with three caps. Each task has 12 clauses and a fresh permutation of the public integer weight multiset  $\{1, \dots, 12\}$ . The regimes are independent clauses, short chains, stars, diamonds, a bridge-connected chain and a rewired connected graph. Clause order changes; the terminal task identities are distinct. Each arm therefore faces 864 registered tasks, and the complete four-arm experiment contains 3,456 terminal opportunities.

The windows preserve the same joint regime/cap mix. This avoids the first study’s confound in which the number of clauses and requirement mix changed over generations. Different weights still alter the optimal answer. The connected regimes require exact fallback rather than factorization into small reusable components. Repeated motifs provide a declared opportunity for inheritance; their frequency is part of the constructed benchmark and must not be represented as observed demand.

Development and qualification instances are separate from terminal instances. Their generator, seeds and partition offsets are frozen. Route selection sees the current source structure and separate development costs, never terminal answers. All permitted construction actions are available at launch; the experiment does not release better algorithms in later windows to manufacture a rising curve. The implemented grammar is closed and authored, and all arms have the same task information and source-bound tools.

The cumulative primitive-unit cap is 20 million per stream and arm. It was selected prospectively to permit a complete comparison over the registered horizon rather than make the deliberately slow frozen instrument the strongest comparator. Actual budget exhaustion, invalid evidence, expired local permission and explicit STOP terminate a stream without replacing missing tasks. The remaining registered tasks score zero. WAIT is a logical due-work condition in this finite experiment, not evidence that a background service remains alive.

| Arm                | Available means and policy                                                                                                                                                                                                                          |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Flywheel           | On first scope, profile bounded and factorized methods on development data, charge both and their independent checks, qualify the selected method, then retain accepted route records and actual component bases. Recheck every current answer.     |
| Conventional cache | The same method grammar, information, adaptation policy, persistence, budgets and current checks. This is an explicit equivalence control; a tie refutes unique superiority.                                                                        |
| Authored expert    | Immediately use the prewritten graph-aware route: factor at cap three when every weak component is at most four nodes; otherwise bounded enumeration. Charge construction and all current checks. This is software, not a timed human-expert trial. |
| Frozen instrument  | The same workbench with full-mask production and verification and no acquired representation. This is a causal intervention comparator, not the strongest conventional workflow.                                                                    |

**Table 3:** Fair access to transformations. The expert’s authored policy and the cache’s reuse ability prevent an artificially forgetful baseline from carrying the main conclusion.

The adaptive scope is defined by current component sizes and cap. Its coarse structure is deliberate: a rule learned for one motif can transfer imperfectly to another with the same sizes. The archive can retain at most 12 bases and 18 route records. Deterministic retirement is implemented, but an unused retirement rule is not counted as observed primary eviction. Every asset has a source identity, exact topology binding and empty inherited proof and authority.

## 3.2 Count what was done, retain what was spent

The primary cost is a transparent structural proxy: a counted primitive visit, test, comparison, arithmetic operation or declared call contributes one unit. The published vector preserves primitive kinds and categories. Byte reads, hashing and serialization are reported separately. The proxy is not an equivalence between JavaScript instructions, Python instructions, electricity and money. Framework and serialization work appears in measured process time but is not exhaustively decomposed into primitive counts.

Acquisition includes losing profiles and candidate qualification. Execution includes production and current verification. Independent Python checking occurs before acceptance, uses the original requested source and cap, and receives the remaining allowance. Its counted work, including work consumed by refusal, is charged. Routing, maintenance and retention remain visible. Restarts and ancillary experiments have separate additional costs rather than being concealed inside a favorable online-only total. The task-order and arm-order schedule is fixed; timing never changes a policy after observation.

The paper reports primary coverage, full counted cost, cost per qualified output, primitive/category vectors, bytes, actual wall/CPU time and bounded load behavior. Unknown authoring, energy and commercial costs remain unknown. A smaller proxy value is a scoped computational finding. It cannot by itself establish lower end-to-end economic cost.

### 3.3 Predeclared acceleration and causal tests

For stream  $s$  and window  $g$ , let

$$R_{s,g} = \frac{Q_{s,g}^{\text{fly}}/C_{s,g}^{\text{fly}}}{Q_{s,g}^{\text{frozen}}/C_{s,g}^{\text{frozen}}}. \quad (8)$$

The bounded productivity-growth criterion requires the mean of  $R_{s,3}, R_{s,4}, R_{s,5}$  to exceed the mean of  $R_{s,0}, R_{s,1}, R_{s,2}$  in all eight streams. This can be passed by acquisition amortization and is not called literal acceleration. The stricter criterion requires, in every stream,

$$\overline{\Delta^2 R_s} = \frac{1}{4} \sum_{g=2}^5 (R_{s,g} - 2R_{s,g-1} + R_{s,g-2}) > 0, \quad R_{s,5} - R_{s,3} > R_{s,2} - R_{s,0}. \quad (9)$$

A plateau following the first window can fail this test even with a large useful speedup. The eight streams, not the individual clauses or documents, are the comparison units. The report exposes each trajectory, range and paired sign rather than pretending thousands of synthetic rows provide thousands of independent replications.

Archive removal permits reconstruction and charges it; it does not make the control intentionally incapable. A replacement test constructs a semantically equivalent basis by a separate implementation and checks fresh output. Changed dependencies, missing masks, false frontier values and current-source mismatches test whether apparent reuse relies on invalid evidence. A separate construction-chain criterion requires a prior means to reduce the cost of producing a genuinely different later means and the saving to support further useful construction. Ordinary task-answer caching does not satisfy it. Any absence or failure of that chain remains visible.

The lower-budget coverage plots are descriptive projections of the realized cost prefixes, explicitly labeled. They are not fresh policy runs at altered budgets. Full task-level observations and all excluded development failures accompany the package.

### 3.4 Freeze and review chronology

The prospective freeze was created at 19:06:50 UTC on 24 October 2026, before the primary run. Its exact 18-file source binding is 39f376fc6374b1888f1fbc4c73b4bed25ef7ca8de62171bec982144c5ace6fa6. The frozen unexecuted archive contains 215 files and has SHA-256 76a267a5db4003b2c922e300e8feb58b43325e24c142b25b400bbcaf2b5eefca. It was saved before the primary launch. Full protocol/task/source identities and the prospective review are supplied with the companion.

The frozen manifest binds the protocol, actual task bytes, producer, controller, task generator, independent checker and unchanged upstream modules. A saved unexecuted package precedes primary execution. A same-session collaborator reviews the numerical and receipt checker independently of the JavaScript producer. Source-level differences after a freeze require an explicit amendment or a new prospective run; old bytes and failures are retained. The final package’s reproduction is a check of the frozen computation, not a new blinded sample.

## 4 The original finding is preserved

Research edition01 remains intact in `preserved/AGI_ALPHA_FLYWHEEL_Complete_Source.zip`, including its authoritative PDF, protocol and raw results. Its 1,152 terminal evaluations used a different task schedule and tariff. The following results are historical motivation, not data pooled into the extension:

| Original workflow  | Qualified / 288 | Declared units | Units / qualified |
|--------------------|-----------------|----------------|-------------------|
| Flywheel           | 271             | 2,017,944      | 7,446.29          |
| Conventional cache | 271             | 2,017,944      | 7,446.29          |
| Fixed selector     | 58              | 1,412,176      | 24,347.86         |
| Authored expert    | 288             | 2,134,383      | 7,411.05          |

**Table 4:** Preserved edition01 results. The expert qualifies more tasks and has slightly lower cost per qualified result. The extension does not erase that finding [1].

In the original stream-zero archive intervention, retention qualifies 36/36 at 316,616 units, while clearing and rebuilding qualifies 30/36 at 279,432. The six additional accepted results cost 37,184 additional units. Reuse saves 12,266 units in one generation, but rebuilding saves 39,662 at equal qualification in another. The original rejected descendant, annotation failure, expensive recovery and cache tie are retained. Changing the production/checking representation is motivated by those results; it is not a retrospective redefinition of their success.

## 5 Results: a useful means can plateau

### 5.1 Complete coverage; different cost to obtain it

All 3,456 registered terminal evaluations qualify: 864/864 for each arm, including every 108-task complete stream. The 20-million-unit per-stream cap does not bind in the primary. Thus the registered coverage endpoint is a tie, not an expansion over the strongest conventional workflow. The meaningful secondary distinction is the cost of obtaining the same qualified structural outputs.

Flywheel and conventional cache each consume 2,887,585 counted units. The authored expert consumes 2,117,413, saving 770,172 units, or 26.67%, against either adaptive arm. The frozen full-mask instrument consumes 93,849,181. The transformation therefore reduces counted work relative to that fixed intervention control by 96.92%, a 32.50-fold ratio. This is not a 32.50-fold inheritance effect, measured CPU speedup, financial return or advantage over the best conventional method.

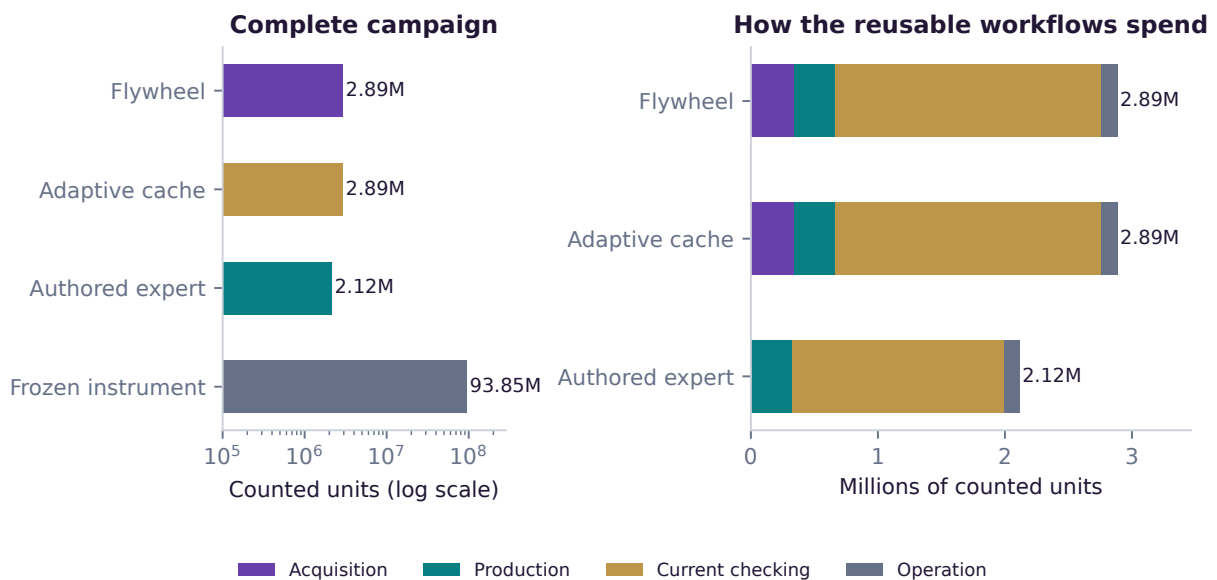


**Figure 2:** All workflows qualify the same registered tasks. Left: total declared primitive units, with a logarithmic axis. Right: the reusable workflows' full counted category breakdown. The expert avoids adaptive profiling overhead. Bytes and unmeasured labor are not converted into these units.

The cost vector explains the gap. Flywheel profiling and qualification account for 251,958 and 84,830 units; construction adds 7,448. Its independent audits consume 1,601,411 units and current JavaScript verification another 485,226. The expert performs the same 325,242 terminal execution units and 485,226 verification units, but avoids adaptive acquisition and its associated audits. Both the benefit and the overhead of deciding what to retain are visible. The conventional cache's exact equality follows from its identical policy, not two independently invented methods coincidentally converging.

The observed recurring-run interval is 72.60 seconds, including oracle waiting and per-window pack persistence, but excluding the final bulk aggregate exports and earlier authoring. Node reports 65.95 seconds user CPU and 5.11 seconds system CPU; those counters exclude Python child CPU. The host is shared with the authoring environment, so these are descriptive observations rather than isolated hardware benchmarks. Original result storage before summary serialization is 187,018,508 bytes. This evidence cost is distinct from the much smaller active capability archive.

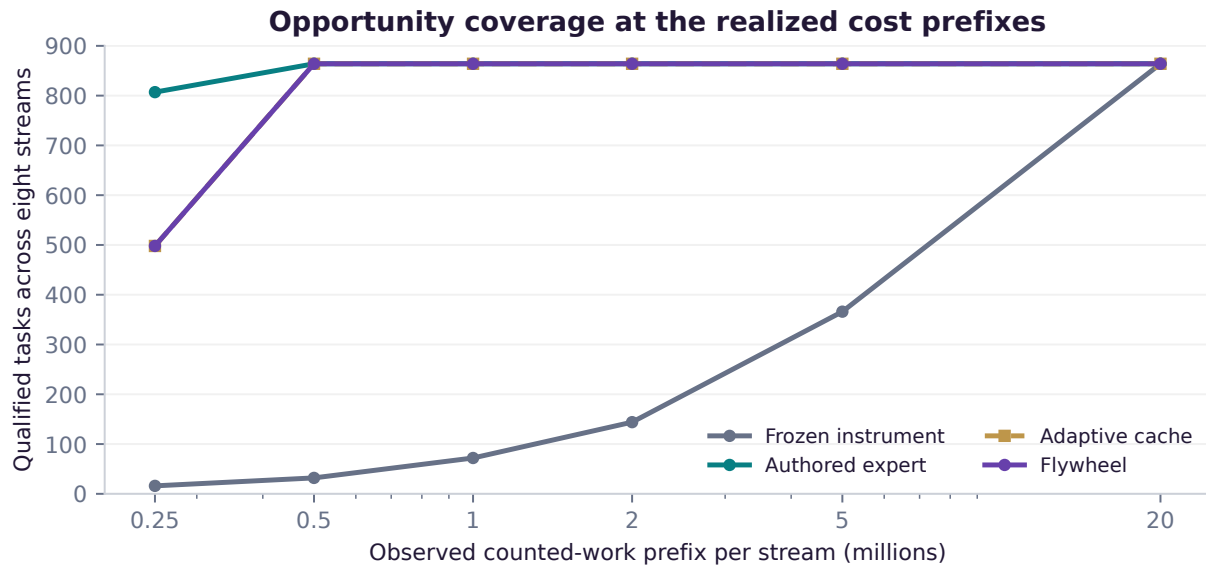


**Figure 3:** Observed budget-prefix projections. At 250,000 units per stream, the realized prefixes contain 498 flywheel/cache, 807 expert and 16 frozen-instrument qualified outputs. At 500,000, the three reusable workflows all contain 864. These are projections of this fixed-policy run, not newly executed policies at altered caps.

## 5.2 What actually accumulates

Each stream retains four distinct component bases and 12 route records by the end of its first window. The four basis contents are a singleton, a three-node chain, a four-node star and a four-node diamond; they respectively contain 2, 4, 8 and 5 admitted feasible masks across cardinalities zero through three. Canonical identical content has the same digest across streams: these are four distinct kinds of executable asset, not 32 newly invented tools.

The final active basis archive is 1,753 canonical bytes per adaptive stream, excluding route records and historical evidence. One additional basis per stream is constructed during terminal work in the first window; other admitted bases come from charged development or qualification work. No new basis appears in windows two through six. Those later windows execute current weighted frontiers on previously admitted bases. There are 22 component bindings per stream per later window, but only four factorized terminal tasks in that window; counting each binding as a separate completed mission would inflate the result.

All first-window acquisition work is included. A typical first window costs about 140,000 units; later windows cost about 44,000. The system has learned a finite routing choice and retained useful bases. It has not kept improving its production method at every window. No primary retirement occurs because the four bases fit within the cap; a separate pressure test exercises retirement.

## 5.3 Causality: method replacement is larger than persistence

The registered removal intervention clears the basis archive before each window while retaining route knowledge and permitting exact reconstruction. On stream one, retention uses 360,179 units and clearing/rebuilding uses 362,174; both qualify 108/108. The 1,995-unit difference is 0.551% of reconstruction cost. It first appears after the next window and persists through the registered horizon. This is a small causal benefit from basis persistence in this stream, not an estimate of the entire acquisition-and-routing package or a population estimate across arbitrary documents.

Reconstruction remains available. Without that allowance, removal could simply cripple the control. Here the lost basis can be rebuilt, so the small difference reveals how much the stored representation contributes. The larger 96.92% contrast against the full-mask instrument is primarily the effect of changing the exact method, a conventional optimization offered to every strong comparator.

A second replacement test constructs a complete chain basis independently in Python and supplies it to the JavaScript factorized producer with construction disabled. Fresh selection and optimum match the retained original; zero JavaScript assets are compiled. Including Python construction, replacement costs 4,098 units versus 3,540 for warm reuse. The canonical asset digest matches because the independently constructed

| Window | Active means                                                                      | Reuse across windows | Counted units |
|--------|-----------------------------------------------------------------------------------|----------------------|---------------|
| 1      |  | 0 / 18 tasks         | 140,048       |
| 2      |  | 4 / 18 tasks         | 44,015        |
| 3      |  | 4 / 18 tasks         | 44,080        |
| 4      |  | 4 / 18 tasks         | 44,019        |
| 5      |  | 4 / 18 tasks         | 44,009        |
| 6      |  | 4 / 18 tasks         | 44,008        |

Stream 1: initial acquisition can be followed by useful reuse and a plateau.

**Figure 4:** An actual stream’s inheritance, with initial acquisition distinguished from later-window reuse. The archive becomes useful quickly and then stops expanding on this finite motif family. New weights create fresh answers, not new algorithms or six distinct successor inventions.

complete representation has identical content. That agreement shows the usable representation, rather than an ancestry label, mediates the result. It does not show independent algorithm discovery.

#### 5.4 The acceleration criterion fails

All eight streams pass the weaker matched-window productivity-growth criterion: amortizing initial acquisition raises later average productivity. All eight fail the stricter acceleration rule. Mean second differences range from  $-7.637$  to  $-7.603$ ; later gains are near zero after the initial jump. The useful improvement plateaus.

| Stream | Flywheel work | Expert work | Early gain | Late gain | Mean $\Delta^2 R$ |
|--------|---------------|-------------|------------|-----------|-------------------|
| 1      | 360,179       | 264,482     | 30.395     | 0.011     | -7.615            |
| 2      | 360,584       | 264,724     | 30.440     | 0.001     | -7.603            |
| 3      | 361,051       | 264,664     | 30.480     | 0.014     | -7.625            |
| 4      | 361,717       | 264,657     | 30.539     | 0.003     | -7.637            |
| 5      | 361,010       | 264,837     | 30.437     | 0.033     | -7.611            |
| 6      | 361,178       | 264,730     | 30.462     | 0.018     | -7.628            |
| 7      | 361,474       | 264,692     | 30.520     | -0.035    | -7.632            |
| 8      | 360,392       | 264,627     | 30.478     | -0.089    | -7.619            |

**Table 5:** Eight paired experimental units. Every stream qualifies 108/108 in both arms. The cache equals the flywheel. Gains and second differences use the frozen normalized-productivity definition, not wall-time ratios.

No multi-asset savings-reinvestment construction chain was included in the executed protocol. Consequently, self-acceleration of capability construction is not demonstrated; it is not inferred from cheaper terminal tasks. This is an explicit unmet part of the ambitious hypothesis. The finished contribution is a runnable loop and a falsifiable decomposition of three effects: replacement of the method, persistence of a useful representation, and saturation of further improvement.

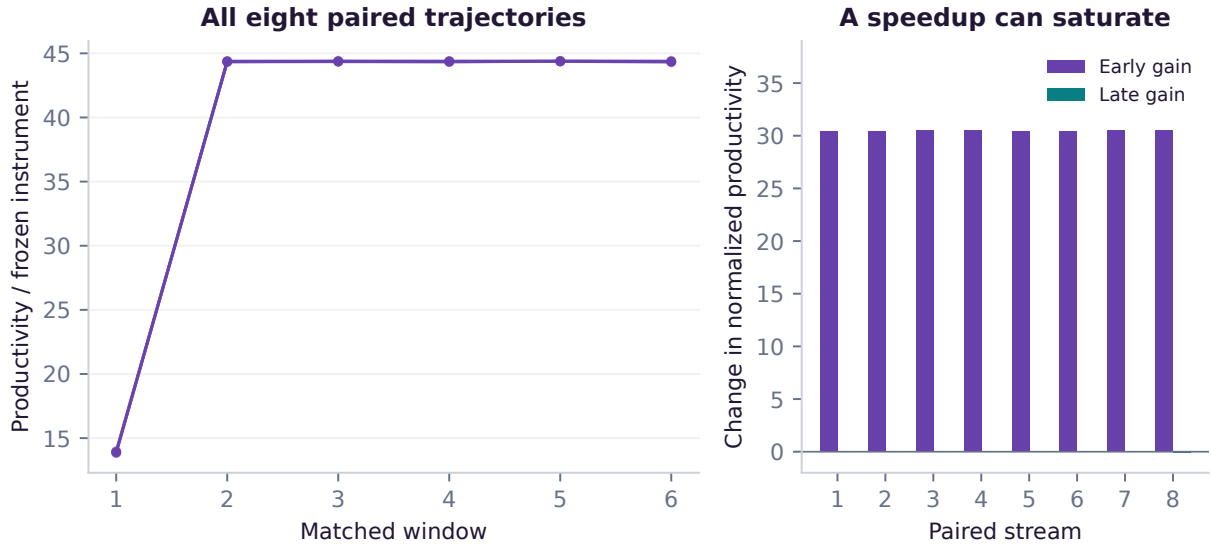


**Figure 5:** Every complete-stream trajectory is shown. The ratio rises after the acquisition window, then stays nearly flat. The right panel contrasts the predeclared early and late gains. This is amortization with saturation, not a rising rate of improvement.

## 6 Recovery, intervention and the cost of staying useful

All 19 registered operating checks pass on the frozen source. They include actual separate-process restart, lost acknowledgement, duplicate/conflicting requests, changed vendored bytes, invalid receipts, false self-reports, incomplete bases, forged frontiers, exhausted budget, invalid/expired/midrun-expired local permission, interruption, archive pressure, waiting and rejected-result restoration.

Lost-acknowledgement recovery and an identical duplicate request each reuse one committed result with zero new commits, but each semantic replay costs 16,514 units. The actual process restart has the same charged replay behavior. The two-unit case stops at exactly two spent units. Midrun expiry preserves 12,941 units of uncommitted work and does not accept a result. The FIFO pressure fixture qualifies 18 tasks while retaining 12 bases and retiring six. Those retirement observations belong to the pressure test, not the primary experiment.

| Node + Python | Qualified | Tasks / second | p50 latency (ms) | p95 latency (ms) |
|---------------|-----------|----------------|------------------|------------------|
| 1 + 1         | 24/24     | 18.13          | 46.99            | 70.19            |
| 2 + 2         | 24/24     | 28.10          | 55.23            | 93.69            |
| 4 + 4         | 24/24     | 35.94          | 89.01            | 122.56           |

**Table 6:** Finite actual-process load: 24 tasks at each concurrency level. Batch elapsed time includes process startup; task latency excludes queue wait. The median uses the frozen upper-middle convention and p95 is nearest-rank.

Increasing this bounded load raises throughput from 18.13 to 35.94 tasks/second while p95 latency rises from 70.19 to 122.56 ms. Qualification remains 24/24, and no human intervenes within a batch. These measurements expose the latency tradeoff rather than establishing general scalability. Every worker and oracle exits after the finite batch; no operating service is left running.

### 6.1 Durable results are not exactly-once computation

The research driver reconciles a request identity with a committed result. A lost acknowledgement must not create a second accepted output. A request with the same identity but changed content must be refused. Recovery checks the frozen source and the task’s semantic receipt, then restores only permitted historical artifacts. Rejected candidates stay rejected. Pure work completed before a durable commit may need to execute again; that work is additional cost, even when no duplicate result appears. The evidence records the actual scope of each fault rather than equating a mocked error flag with process recovery.

This distinction is familiar from durable operation protocols: suppressing duplicate externally visible results

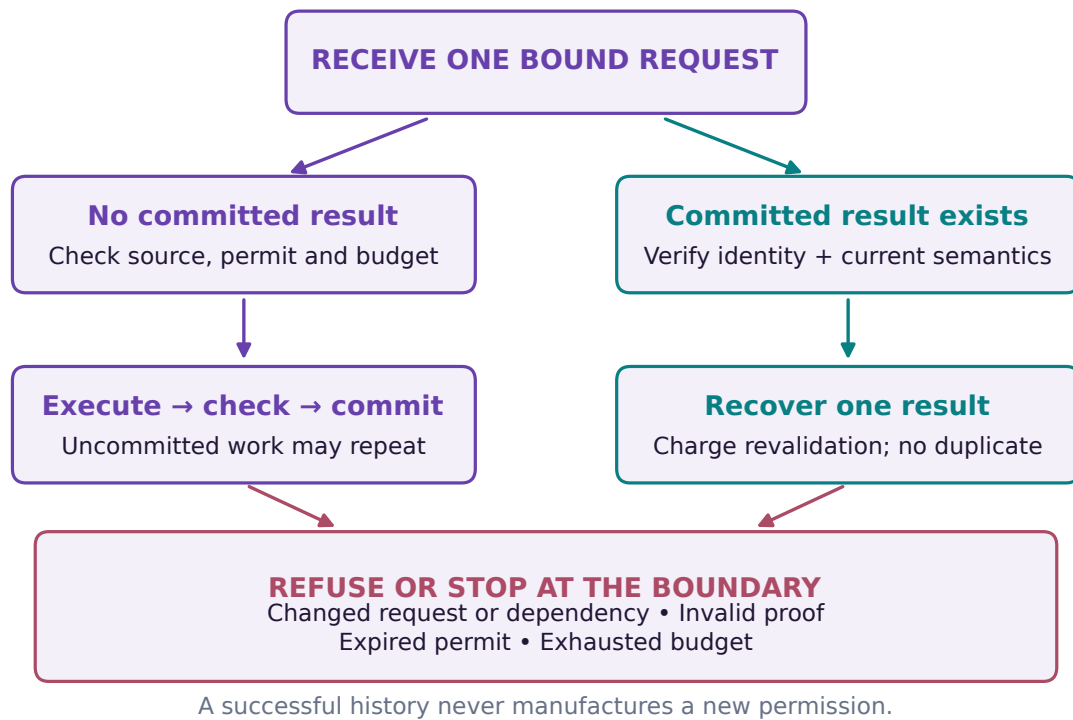


**Figure 6:** Recovery preserves one committed result while charging verification and any repeated pure computation. Failed current proof or permission cannot be repaired by importing an old success.

requires identity and result tracking [13]. The present local study does not claim a distributed transaction system, consensus, exactly-once CPU execution or exactly-once external settlement. All external effects remain disconnected.

Successful operation includes refusal. The loop can return WAIT when no permitted request is due, STOP when its resource or local permit is exhausted, retain an incumbent when a candidate adds no benefit, and fall back to exact reconstruction when a dependency graph no longer fits a retained basis. Current authority is checked at the operating boundary; an old successful record does not extend an expired permit.

## 6.2 An intervention ledger with unknowns left visible

The experiment's autonomy denominator begins at the one bounded primary launch. After that launch, the finite driver selects the next manifest entry and advances its phases. This is narrower than autonomous project creation. A person supplied the commission; assistant collaborators selected the mission family, designed the representation, wrote and debugged the code, inspected failures, froze the protocol, launched the study, reviewed outputs and typeset the paper.

The authoring ledger records those activities separately from within-run interventions. It records counts and timestamps where observed. Human review minutes, historical design hours, model inference used for authoring, energy, hardware amortization and long-term production upkeep are unknown. The study therefore supports computational comparisons under its declared accounting, not a complete economic profit claim. An independently written same-session checker is a useful separation of implementation; it is not an independent institution or a blinded external reviewer.

Evidence itself has a cost. Active reusable assets and cumulative receipt history are reported separately. Bounding the active archive does not bound the history forever. The finite load experiment measures actual worker processes, elapsed batch time, latency, qualification and recovery within the declared limits. It does not justify a claim about a continuously operating enterprise or millions of simultaneous users.

## 7 WORLD, POLICY, PROOF and AUTHORITY

| Surface   | What is inspectable and what can carry forward                                                                                                                                     |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WORLD     | The exact source clauses, prerequisite graph, weights, current cap, task identity and dependency bytes. A changed world requires fresh applicability checks.                       |
| POLICY    | Frozen method grammar, routing/retention rule, budgets, due-work order, exclusions and stop conditions. Candidate data cannot edit these through its interface.                    |
| PROOF     | Historical receipts and their provenance may be inherited. Current graph, basis completeness, objective and certificate are checked anew. No favorable self-report is final proof. |
| AUTHORITY | A finite local research permit allows this execution. Assets and restored packs carry no production grant, no spending permission and no settlement authority.                     |

**Table 7:** Inheritance transfers means and provenance. It does not transfer present-tense truth or permission.

The released workbench records `independentProof:false`, admission `HOLD`, empty authority and settlement `NOT_CONNECTED`. Its pack inspector establishes structural integrity and binding, not the semantic truth of a result. The companion's mathematical audit adds a scoped research judgment without converting that judgment into production admission. A candidate cannot acquire permission by changing its own favorable report.

The maintained native runtime, its signed Journal and cycle commands, the preserved `AgentManager` research scheduler, the workbench adapter and this finite source experiment are distinct surfaces. The study uses the eight vendored modules pinned to Alpha commit `787bc547`; it does not initialize native credentials. The separate native operating owner continues the installed `CampaignExperiment/DiscoveryLoop` lane. At the inspected released source, `CampaignExperiment` fixes archive access to `none`; running that cold study repeatedly would not establish the retained-program result investigated here [14].

The ordinary source merge of PR 4890 during preparation advanced main to `aedc9277`. It did not alter this experiment's frozen source or clear an original Harness/Fleet restriction. Exact publication, installed-runtime acceptance and a verified public deployment remain separate evidence. The source audit and publication handoff preserve those boundaries.

## 8 Applicable interfaces and \$AGIALPHA utility

The empirical loop is zero-funded software execution. The four-repository and five-contract maps below explain relevant source functions; they do not imply that contracts or adapters were exercised. Exact inspected commits and extracted source are in `audit/FLYWHEEL_SOURCE_AUDIT.md` and its manifest.

| Repository       | Relevant inspected interface                                                                                                | Evidence level                        |
|------------------|-----------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| AGI Alpha Agent  | Credential-free workbench planning, bounded execution, result packs and history; native orchestration inspected separately. | Workbench executed; native inspected. |
| AGI Jobs Manager | <code>manager-usdc/USDC6</code> work-request inspection, expected bindings and semantic-inspector seam.                     | Source inspection.                    |
| AGI Jobs v0      | <code>jobs-agialpha/AGIALPHA18</code> work-request receiver and separately scoped inspection.                               | Source inspection.                    |
| AGI Fleet Linux  | Exact native-source inventory and admission boundary for separately qualified execution.                                    | Source inspection.                    |

**Table 8:** The receivers advertise local inspection, unprovisioned external execution and no submission authority. A diagram of four repositories is not an installed four-product acceptance test [14–17].

Practical \$AGIALPHA utility would concern collateral, stake, budgeted funding and job/reviewer settlement in the applicable authorized route. Manager's USDC path remains distinct. No token is required to reproduce the present research. No local EVM, simulated settlement, installed financial adapter, live balance/authority readback or token transfer was performed. Issuance, scarcity, a burn mechanism or price appreciation is not a measurement of productive value.

| Alpha contract      | Relevant source function and conditional application                                                                                                                                   |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AscensionAccess     | owns, admitted, eligible: live ENS ownership and expiring admission can constrain who may participate in a separately authorized role.                                                 |
| NovaSeed            | seal, plan: capsule, FusionPlan, evidence and optional parent commitments bind a declared artifact and lineage. Ownership is not a quality verdict.                                    |
| AscensionRiskOracle | lockForJob, attest, green: stake locking and expiring quorum attestations can provide a separate risk gate. Distinct addresses do not establish institutional independence.            |
| AscensionJobMarket  | postFromSeed, submit, validate, claimRefund: bounded jobs, escrow, delivery-bound review and refunds can settle authorized work. Settlement is distinct from the mathematical checker. |
| AscensionMark       | open, route, collect, close: funding, plan-constrained routing, horizon/budget limits and capital return can finance a separately permitted campaign.                                  |

**Table 9:** All five contracts were inspected at the frozen Alpha source. None was executed by this experiment.

## 9 What the experiment cannot establish

**Annotation truth.** Exact optimization can faithfully preserve a false or incomplete input graph. The original flywheel’s constructed omission challenge achieves structural score 20 while the alternate-truth score is 0. The preferred Harness detector paper supplies an analogous identification boundary: the same five readings can fit mean signal 100 under unit gain or mean 92 under gain 25/23. Its reported interval [98, 102] misses 92. The earlier paper’s 96/96 conditional qualifications, 78/96 threshold decisions, cache tie and gain-error failure remain unchanged [1, 18].

**Scientific invention.** The system searches and applies a closed supplied grammar. The architecture enables inspection of what is retained, but this does not establish the discovery of open-ended science, a general autonomous research capability or new physical means of production.

**Generalization.** Fresh task instances avoid literal answer reuse. Their author-designed generator and same-session custody do not make them a natural population or an externally blinded benchmark. Every complete stream is the comparison unit. Uncertainty is conditional on this finite generator and the declared schedule. Repeated exploration of terminal tasks would invalidate confirmatory interpretations; this is why development, freeze, confirmation and later ancillary analysis are separately labeled [19].

**Economic and operational growth.** Computational units are neither currency nor elapsed seconds. Unknown labor can reverse an apparent computational advantage. The driver stops at its declared boundary, and the browser runs only while open. No outreach, customer demand, revenue, continuously running service or production self-modification is demonstrated.

### The standard survives a negative result

Prove that the loop runs. Test whether capability compounds. Show when it does not. A useful accepted capability earns its place through fresh work; its lineage is provenance, not a substitute for a new result.

## 10 Reproduce the experiment and inspect the inheritance

### 10.1 Start with a useful browser result

Open `demo/AGI_ALPHA_FLYWHEEL_Acceleration_Demo.html` and choose `Try it now`. The standalone English/French interface executes a separate six-mission stream through the unchanged released workbench and the acceleration core: build a useful basis, reuse it under changed weights, construct a specialization, reuse it, refuse incompatible structure and return to compatible retained means. The output is an actual short synthetic brief and an inspectable executable basis, not an animation of a claimed service.

The fresh view includes all three comparator computations in its total cost counter. A reader can stop after a checkpoint, export it, reopen the page, restore it with charged semantic rechecks, and resume unfinished mission identities. The page also exports the brief and the reusable asset. Current proof and production authority are visibly distinct. Boundary probes change copies of data and preserve the original result.

The recorded-study tab displays the immutable primary summary. It is explicitly precomputed evidence. Fresh in-tab computation uses the JavaScript structural checker; it does not run the independent Python oracle. The browser teaching sample is not substituted for the 3,456-evaluation confirmation. Closing the tab stops computation. The interface needs no account, installation, wallet, payment, API key or external request.

The final HTML passes the packaged actual-Chromium checks for fresh work, inheritance, incompatibility, downloads, stop/restore/resume, tamper refusal, keyboard navigation and narrow 320/390-pixel English/French views. Its QA report records the check count and binds the exact tested HTML hash. This establishes local browser behavior, not a verified public website deployment.

### 10.2 Reproduce the research computation

The tested core environment is Node.js 24.22.1 and Python 3.12.14, using only their standard libraries. No model provider or network is required after the package and runtimes are present. From the extracted complete package:

```
node companion/run.mjs reproduce fresh-run
node companion/run.mjs faults fresh-faults
node companion/run.mjs load fresh-load
node companion/run.mjs ablations fresh-ablations
```

Each destination must be new. The reproduction command checks source/protocol/task hashes, runs the complete fixed stream and stops. Ancillary runs use declared fixtures; they do not retrain on terminal outcomes. The archive intervention reads the preserved original packs as its registered reference. Workbench timestamps and pack digests may differ on reproduction; objectives, selections, costs, task identities and source bindings must agree.

A fresh extraction of the staged runnable archive reproduces all 3,456 observation rows byte-for-byte. All 192 window accounting records and workbench scientific payloads agree after the explicit timestamp/pack-identity exclusions. The independent post-run audit passes 30,950 consistency checks. Supplemental QA work is recorded separately: two audit passes consume 10,839,808 oracle units plus 105,600 integrity units. These reporting costs are not hidden inside treatment costs or presented as zero authoring overhead.

### 10.3 Rebuild the paper and its original vector figures

The complete compilable source includes the manuscript, all section files, bibliography, editable SVGs and vector PDFs. The authoritative PDF is preserved inside both the complete source archive and the compact LaTeX archive. To rebuild in a new directory:

```
python3 paper/build_paper.py --output rebuilt-paper
```

The builder uses XeLaTeX and BibTeX with TeX Live 2023, fixed source timestamps and disabled shell escape. It records the resulting SHA-256 and compares rebuilt bytes with the authoritative PDF. To regenerate figures, use the optional pinned Matplotlib 3.10.8/NumPy 2.3.5 environment and run `paper/make_figures.py`. Figure data comes directly from original results. Numerical tables and result macros are generated by `paper/collect_results.py`.

## 10.4 Evidence, preservation and the publication boundary

| Package surface | What a reader can verify                                                                                                      |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------|
| companion/      | Frozen source and protocol, actual task bytes, original raw outputs, exact workbench packs and independent checker.           |
| review/         | Separate implementation review, development failures, prospective disposition and post-run arithmetic/receipt audit.          |
| evidence/       | Claim docket, intervention ledger, source/runtime identities, result extraction and publication verification.                 |
| demo/           | Self-contained fresh/replay interface, editable source, bundle hashes and actual browser QA.                                  |
| paper/figures/  | Six original editable vector figures and the source generator.                                                                |
| preserved/      | Unchanged original edition, authoritative original PDF, original acceleration design checkpoint and saved prospective freeze. |

**Table 10:** Every material empirical claim has a concrete evidence destination. The exhaustive manifest binds payload paths, sizes and hashes.

The first post-run QA report is preserved alongside its correction: absent zero-count categories were initially compared with explicit zero as different. Only the supplementary reporting comparator changed; the frozen runtime, oracle, protocol and primary results were unchanged. This is a QA representation error, not a favorable replacement experiment.

The additive handoff preserves the original research PR 4893 and the active publication coordinator at issue 4850. It supplies exact new artifact identities without editing a shared catalogue, substituting a release owner or clearing an original Harness/Fleet action. Passive research publication, executable companion review and live served-byte acceptance remain separate steps. No website deployment is claimed by this package.

### Find your Alpha. Build your successor $\Omega$ .

Keep the means. Re-earn the proof. Make the next cycle cheaper where the evidence supports it—and keep the result when it does not.

## References

- [1] Vincent Boucher. AGI ALPHA FLYWHEEL: From completed work to compounding capability, 2026. URL <https://github.com/MontrealAI/AGI-Alpha-Agent-v0/tree/ec2f9ed8644b1d83ea89d6d619e384c321d5949a/studies/agi-alpha-flywheel-v1>. Research edition01, preserved original experiment, commit ec2f9ed8644b1d83ea89d6d619e384c321d5949a.
- [2] Vincent Boucher. Transformational intelligence: Learning to change the game, 2026. URL [https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Transformational\\_Intelligence.tex](https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Transformational_Intelligence.tex). MONTREAL.AI research manuscript, October 2026.
- [3] Vincent Boucher. Out-learn: Change what you need to know, 2026. URL [https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out\\_Learn.tex](https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out_Learn.tex). Constructed Experiments and Sharp Calibration Guarantees for Reusable Learning; revised October 21.
- [4] Vincent Boucher. Out-think: Transformational intelligence for inventing the space of reasoning, 2026. URL [https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out\\_Think.tex](https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out_Think.tex). MONTREAL.AI research manuscript, October 21.
- [5] Vincent Boucher. Out-design: Creating the freedom to transform, 2026. URL [https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out\\_Design.tex](https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out_Design.tex). MONTREAL.AI revised research manuscript, October 21.
- [6] Vincent Boucher. Out-strategise: Transformational intelligence for constructing the conditions of success, 2026. URL [https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out\\_Strategise.tex](https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out_Strategise.tex). MONTREAL.AI research manuscript, October 20.
- [7] Vincent Boucher. Out-execute: Constructing the conditions of completion, 2026. URL [https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out\\_Execute.tex](https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out_Execute.tex). MONTREAL.AI revised research manuscript, October 21.

- [8] Vincent Boucher. Out-evolve: Building the means of discovery and production, 2026. URL [https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out\\_Evolve.pdf](https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/Out_Evolve.pdf). MONTREAL.AI revised research manuscript, October 20.
- [9] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning, 2020. URL <https://arxiv.org/abs/2006.08381>.
- [10] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL <https://arxiv.org/abs/2305.16291>.
- [11] Sayash Kapoor, Benedikt Stroebel, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter, 2024. URL <https://arxiv.org/abs/2407.01502>.
- [12] Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer, 2020. URL <https://arxiv.org/abs/2009.00909>. Revised August 2021; published in IEEE/CAA Journal of Automatica Sinica, 2022.
- [13] Collin Lee, Seo Jin Park, Ankita Kejriwal, Satoshi Matsushita, and John Ousterhout. Implementing linearizability at large scale and low latency. In *Proceedings of the 25th ACM Symposium on Operating Systems Principles*, 2015. doi: 10.1145/2815400.2815416. URL <https://sigops.org/s/conferences/sosp/2015/current/abstracts.html>.
- [14] MONTREAL.AI. AGI Alpha Agent v0: released source and operator documentation, 2026. URL <https://github.com/MontrealAI/AGI-Alpha-Agent-v0/tree/787bc547f8c44854931b316d7286b8bc9f9c8bba>. Version 1.41.1; exact commit 787bc547f8c44854931b316d7286b8bc9f9c8bba; inspected October 24.
- [15] MONTREAL.AI. AGI Jobs Manager: source interface snapshot, 2026. URL <https://github.com/MontrealAI/AGIJobManager/tree/c5958eae872458c2f3cac48fea0b4bbfadc6ac6c>. Exact commit c5958eae872458c2f3cac48fea0b4bbfadc6ac6c; inspected October 24.
- [16] MONTREAL.AI. AGI Jobs v0: source interface snapshot, 2026. URL <https://github.com/MontrealAI/AGIJobsv0/tree/a24258ddef16d108a988c0b94b191ad5f3803ab9>. Exact commit a24258ddef16d108a988c0b94b191ad5f3803ab9; inspected October 24.
- [17] MONTREAL.AI. AGI Fleet Linux: source interface snapshot, 2026. URL <https://github.com/MontrealAI/AGI-Fleet-Linux/tree/45759fb096faf8988f58ae5ea71248d0f35a97c7>. Exact commit 45759fb096faf8988f58ae5ea71248d0f35a97c7; inspected October 24.
- [18] MONTREAL.AI. SUCCESSOR  $\Omega$  harness: Make every mission build what comes next, 2026. URL <https://github.com/MontrealAI/AGI-Alpha-Agent-v0/blob/787bc547f8c44854931b316d7286b8bc9f9c8bba/docs/papers/successor-omega-harness-detector-v1.pdf>. Preferred 28-page detector research edition, October 24.
- [19] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse, 2015. URL <https://arxiv.org/abs/1506.02629>.